CEX Software Manual

Release 6.12

Embention

2024-04-09

CONTENTS

1 Software applications **3**
- 1.1 Veronte Link . . . 3
- 1.2 CEX PDI Builder . . . 3

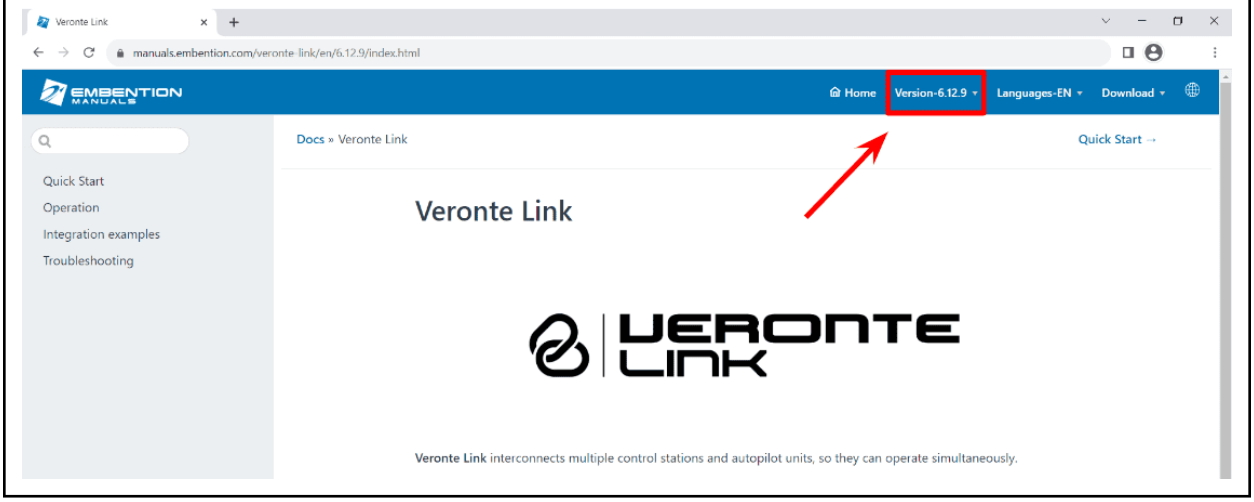
2 Lists of interest **5**
- 2.1 Lists of variables . . . 5
  - 2.1.1 BIT Variables . . . 5
  - 2.1.2 Real Variables (RVar) - 32 Bits . . . 7
  - 2.1.3 Integer Variables (UVar) - 16 Bits . . . 8
- 2.2 List of addresses . . . 8

3 CAN Bus protocol **11**
- 3.1 CEX Status . . . 13
- 3.2 Arbitration . . . 13
- 3.3 Command PWMs . . . 14
- 3.4 Lift MCU telemetry . . . 15
  - 3.4.1 CEX to 1x . . . 15
  - 3.4.2 1x to CEX . . . 16
- 3.5 Scorpion Tribunus ESC Telemetry (Lift) . . . 16
- 3.6 JetiTM ESC Telemetry . . . 17
- 3.7 Jeti BEC Telemetry . . . 17
- 3.8 Jeti Temperature Sensor Telemetry . . . 18
- 3.9 Set Maintenance Mode Command . . . 18
- 3.10 Stick Selection Command . . . 18

In this manual the user can consult a brief description of all the applications created and designed to work together with the Veronte CEX.

In addition, links are available to access the manuals for each of these applications.

Warning: Select your version before reading any user manual for software. The following image shows where to select a version from any Embention user manual.



CHAPTER

ONE

SOFTWARE APPLICATIONS

First of all, Veronte Link is required to connect a CEX to a computer. Then, it can be configured with CEX PDI Builder.

1.1 Veronte Link

Veronte Link establishes communication between a computer and any Veronte product by creating a VCP bridge. It allows to use multiple control stations and autopilots to be interconnected, operating simultaneously. **Veronte Link** also includes a post-flight viewer, to reproduce all recorded data from previous flights and generate plots and reports.

For more information, visit the Veronte Link user manual.

1.2 CEX PDI Builder

CEX PDI Builder is the main configuration tool to adapt a **CEX** to be suitable for a specific system, being its main goal to expand the available communication protocols. **CEX PDI Builder** includes:

- Communications: Through general purpose CAN bus, inputs and outputs and PWMs.
- Stick control signal management: Compatible with **Stick Expander**, Futaba, Jeti, FrSky and TBS. It includes custom configuration for other sticks.
- Arbitration: **CEX** is able to send PWM signals using arbitration in the same way **Veronte Autopilot 4x** does.

Read the CEX PDI Builder user manual for more details.

Note: By default, **CEX** has not any configuration. In consequence, **CEX** will be in maintenance mode and **Veronte Link** will show the **Loaded with Error** status. Nonetheless, it is possible to load a new configuration with **CEX PDI Builder**; since the maintenance mode allows to connect a computer and load any configuration, with any connection (USB, RS-232, RS-485 or CAN).

CHAPTER

TWO

LISTS OF INTEREST

2.1 Lists of variables

This section shows all the variables employed by **Veronte CEX**.

2.1.1 BIT Variables

ID	Name	Description
6	File system error	System file manager
7	System error	This bit checks whether the system is running properly. 0 for system error, 1 for system OK.
8	Memory Allocation	RAM allocation - 0 for trying to use more than available memory, 1 for running
9	PDI error	PDI files - Dependent on *PDI Error Source (UVar 50)* • 0 for wrong PDI configuration: if *PDI Error Source* > 0 • 1 for running OK: if *PDI Error Source* $== 0$
12	System power up bit error	Power up - 0 for error, 1 for OK
16	Stack core 1 usage FAIL	0 for stack overflow, 1 for OK
60-64	Sensor-External I2C device 0-4	External communication I2C from device 0 to 4
65	SCI A Transmitting (Sara)	Serial Communication Interface - sara transmission
66	SCI A Receiving (Sara)	Serial Communication Interface - sara reception 0 for not receiving, 1 for receiving
67	SCI B Transmitting (Radio)	Serial Communication Interface - radio transmission
68	SCI B Receiving (Radio)	Serial Communication Interface - radio reception 0 for not receiving, 1 for receiving

continues on next page

Table 1 – continued from previous page

ID	Name	Description
69	SCI C Transmitting (RS485)	Serial Communication Interface - RS485 transmission
70	SCI C Receiving (RS485)	Serial Communication Interface - RS485 reception 0 for not receiving, 1 for receiving
73	CAN A ERROR	CAN A state - 0 for error, 1 for OK
74	CAN B ERROR	CAN B state - 0 for error, 1 for OK
75	CAN A warning	CAN A state - 0 for warning, 1 for OK
76	CAN B warning	CAN B state - 0 for warning, 1 for OK
96-98	SCI A-C receiving error	SCI A to C - 0 for error in this port (invalid format or configuration), 1 for OK
102-103	CAN A-B receiving	CAN A to B communication - 0 for not receiving, 1 for receiving
104	Stick PPM 0 not detected	Stick PPM 0 - 0 for not detecting, 1 for detecting
111-112	CAN A-B transmitting	CAN signals A to B - 0 for not transmitting, 1 for transmitting
120-123	Pulse 0-3 not detected	Pulse 0 to 3 detection - 0 for pulse not detected, 1 for detected
330	Jetibox COMM Error	0 for error with Jetibox communications, 1 for Jetibox communication OK
800-811	PWM 0-11 GPIO Off	PWM GPIO 0-11 communication State - 0 for Off, 1 for On
816-819	EQEP_A-I (GPIO17-20) Off	Input/Output State - 0 for Off, 1 for On
823	GPIO 5 (GPIO28) Off	GPIO 5 Status (Low/High) - 0 for Off, 1 for On
824	GPIO 6 (GPIO61) Off	GPIO 6 Status (Low/High) - 0 for Off, 1 for On
825	GPIO 8 (GPIO60) Off	GPIO 7 Status (Low/High) - 0 for Off, 1 for On
826	GPIO 8 (GPIO59) Off	GPIO 8 Status (Low/High) - 0 for Off, 1 for On
827	GPIO 9 (GPIO17) Off	GPIO 9 Status (Low/High) - 0 for Off, 1 for On
828	GPIO 10 (GPIO58) Off	GPIO 10 Status (Low/High) - 0 for Off, 1 for On
829	GPIO 11 (GPIO16) Off	GPIO 11 Status (Low/High) - 0 for Off, 1 for On
830	GPIO 12 (GPIO53) Off	GPIO 12 Status (Low/High) - 0 for Off, 1 for On
831	GPIO 13 (GPIO20) Off	GPIO 13 Status (Low/High) - 0 for Off, 1 for On
832	GPIO 14 (GPI023) Off	GPIO 14 Status (Low/High) - 0 for Off, 1 for On

continues on next page

Table 1 – continued from previous page

ID	Name	Description
833	GPIO 15 (GPI051) Off	GPIO 15 Status (Low/High) - 0 for Off, 1 for On
834	GPIO 16 (GPI052) Off	GPIO 16 Status (Low/High) - 0 for Off, 1 for On
835	GPIO 17 (GPI049) Off	GPIO 17 Status (Low/High) - 0 for Off, 1 for On
836	GPIO 18 (GPI008) Off	GPIO 18 Status (Low/High) - 0 for Off, 1 for On
837	GPIO 19 (GPI011) Off	GPIO 19 Status (Low/High) - 0 for Off, 1 for On
838	GPIO 20 (GPI010) Off	GPIO 20 Status (Low/High) - 0 for Off, 1 for On
839	GPIO 21 (GPIO09) Off	GPIO 21 Status (Low/High) - 0 for Off, 1 for On
1010-1113	Custom msg 0-103 Rx Error	Custom message timeout - 0 for error, 1 for OK
1200-1499	User BIT 00-299 Error	User bit 0 to 299 - 0 for error, 1 for OK

2.1.2 Real Variables (RVar) - 32 Bits

ID	Name	Units/Values	Description
50	CAN-A Tx Rate	pkts/s	CAN-A transmission packet rate
51	CAN-B Tx Rate	pkts/s	CAN-B transmission packet rate
52	CAN-A Tx skip Rate	pkts/s	CAN-A messages delayed because no mailbox is available for sending
53	CAN-B Tx skip Rate	pkts/s	CAN-B messages delayed because no mailbox is available for sending
300	Relative Timestamp	s	Time spent since power-on of the system
700-703	RPM 0-3	rad/s	Angular speed associated to pulse captured 0-3
800-807	PWM 0-7	custom type	Pulse Width Modulation signal 0 to 7
1100-1104	Lidar 0-4 Distance	m	Configurable variables for Lidar distances 0 to 4
1320-1321	ADC 3.3V Input 0-1	V	CEX ADC 3.3 V inputs 0 and 1
1322-1323	ADC 5.0V Input 0-1	V	CEX ADC 5.0 V inputs 0 and 1
1324-1325	ADC 12.0V Input 0-1	V	CEX ADC 12.0 V inputs 0 and 1
1326-1327	ADC 36.0V Input 0-1	V	CEX ADC 36.0 V inputs 0 and 1
1328-1329	ADC vIn 0-1	V	CEX External power supplies 0 and 1
1330	PCB Temperature	K	CEX PCB Temperature (from ADC input)
1450-1453	Captured Pulse 0-3	customType	Input values from pulses
3100-3399	User Variable 00-299 (Real - 32 Bits)	customType	Free variables for the user to use

2.1.3 Integer Variables (UVar) - 16 Bits

ID	Name	Description
50	PDI Error Source	Index for PDI error source identification. For further information, consult the List of PDI errors section of the **1x Software Manual**
90	Version Major	Major software version
91	Version Minor	Minor software version
92	Version Revision	Revision software version
95	UAV Address	UAV address
450	CAN-A Tx errors	CAN A communication errors in transmission
451	CAN-A Rx errors	CAN A communication errors in reception
452	CAN-B Tx errors	CAN B communication errors in transmission
453	CAN-B Rx errors	CAN B communication errors in reception
454-455	CAN to Serial 0-1 frames dropped	Lost messages during CAN to Serial transformations
495-496	Global configuration state (crc) of files-memory (Higher-Lower 16 bits)	Global configuration state (crc) of files-memory
498-499	Global configuration state (crc) of files-memory	Global configuration state (crc) of files-memory
600-615	PPM channel 0-15 output	CEX PPM channel outputs
620	Jetibox max successfully parsed message	Maximum Jetibox messages successfully parsed
1000-1299	User Variable 00-299 (Unsigned Integer - 16 bits)	Free variables for user

2.2 List of addresses

Every Embention device communicate with other devices/tools using its address through VCP.

The following list contains all these addresses:

Address	Recognized as	Description
0	Dummy for pdi builders	Dummy for pdi builder
1	Cloud	**Veronte Cloud** address
2	Vlink	Address used by **Veronte Link** app to communicate with Veronte units
2-3	App + *Address*	Veronte applications addresses. **App 2** is the one used by default by Veronte applications, although *App 3* is also available
255-511	App dynamic + *Address*	**Dynamic addresses** for Veronte applications
998	Broadcast	To **all devices** on a network
999	Address unknown	This address can be used for a device that **does not** have a **valid address** configured
1000-1777	1x v4.0 + *Address*	Specific address of an **Autopilot 1x** with **hardware version 4.0**
1778-3999	1x v4.5 + *Address*	Specific address of an **Autopilot 1x** with **hardware version 4.5**
4000-17999	1x v4.8 + *Address*	Specific address of an **Autopilot 1x** with **hardware version 4.8**
18000-19899	1x BCS + *Address*	Specific address of a **BCS** unit
19900-19999	1x v4.7. For internal use only + *Address*	Specific address of an **Autopilot 1x** with **hardware version 4.7**
20000-21999	Smart Can Isolator + *Address*	Specific address of a **Smart Can Isolator** unit
30000-31999	MC01 + *Address*	Specific address of a **MC01** unit
32000-34999	MC24 motor controller + *Address*	Specific address of a **MC24** unit
35000-39999	MC110 motor controller + *Address*	Specific address of a **MC110** unit
40000-41999	CEX + *Address*	Specific address of a **CEX** with **hardware version 1.2**
42000-43999	MEX + *Address*	Specific address of a **MEX** unit
44000-49999	CEX2 + *Address*	Specific address of a **CEX** with **hardware version 2.0**
50000-51089	Arbiter v1.0 + *Address*	Specific address of an **Arbiter** with **hardware version 1.0**
51090-51999	Arbiter v1.2 + *Address*	Specific address of an **Arbiter** with **hardware version 1.2**
52000-59999	Arbiter v1.8 + *Address*	Specific address of an **Arbiter** with **hardware version 1.8**
60000-65535	Reserved + *Address*	Reserved addresses
65536-69631	Virtual v4.0 + *Address*	Specific address of a **Virtual Autopilot 1x** with **hardware version 4.0**
69632-73727	Virtual v4.5 + *Address*	Specific address of a **Virtual Autopilot 1x** with **hardware version 4.5**
73728-77823	Virtual v4.8 + *Address*	Specific address of a **Virtual Autopilot 1x** with **hardware version 4.8**

CHAPTER

THREE

CAN BUS PROTOCOL

This section defines the CEX communication protocol.

This is the configuration of the messages that must be performed in **Veronte Autopilot 1x to communicate with CEX**.

Note: No configuration of these messages is required in CEX, as CEX is already internally configured to "understand" messages configured in this way.

Warning: For these messages sent from the 1x to be processed correctly, they must be received by the 'Consumer' **Application processor**.

CEX Communication Protocol via CAN Bus is defined as follows:

1. **cmd (8 bits - 1 byte)**: First byte refers to the **Message Type**.

   Messages Type are defined as follows:

Type	Value	Description
t_arbitration	0	Arbitration message
t_version	1	Version request / response
t_pwm_0_3_set	2	PWMs 0 to 3
t_pwm_4_7_set	3	PWMs 4 to 7
	4	Reserved
t_esc_tm	5	Scorpion Tribunus ESC telemetry data
t_esc_tm2	6	Jeti ESC telemetry data
t_bec_tm1	7	Jeti BEC telemetry data 1
t_bec_tm2	8	Jeti BEC telemetry data 2
t_temp_tm	9	Jeti Temperature sensor telemetry data
t_mcu_cmd	10	Lift MCU battery command
t_pwm_8_11_set	11	PWMs 8 to 11
t_pwm_12_15_set	12	PWMs 12 to 15
t_pwm_16_19_set	13	PWMs 16 to 19
	14	Reserved
	15	Reserved
t_cmd_maint	16	Command to go to Maintenance Mode
t_stick_sel	17	Command for Stick selection
t_mcu_tm1	18	Lift MCU telemetry data 1
t_mcu_tm2	19	Lift MCU telemetry data 2

Note: All these *Message Type* are defined as a "Matcher" in the CAN custom messages configuration. For example, for PWMs 0-3, the *Message Type* will be configured as follows:

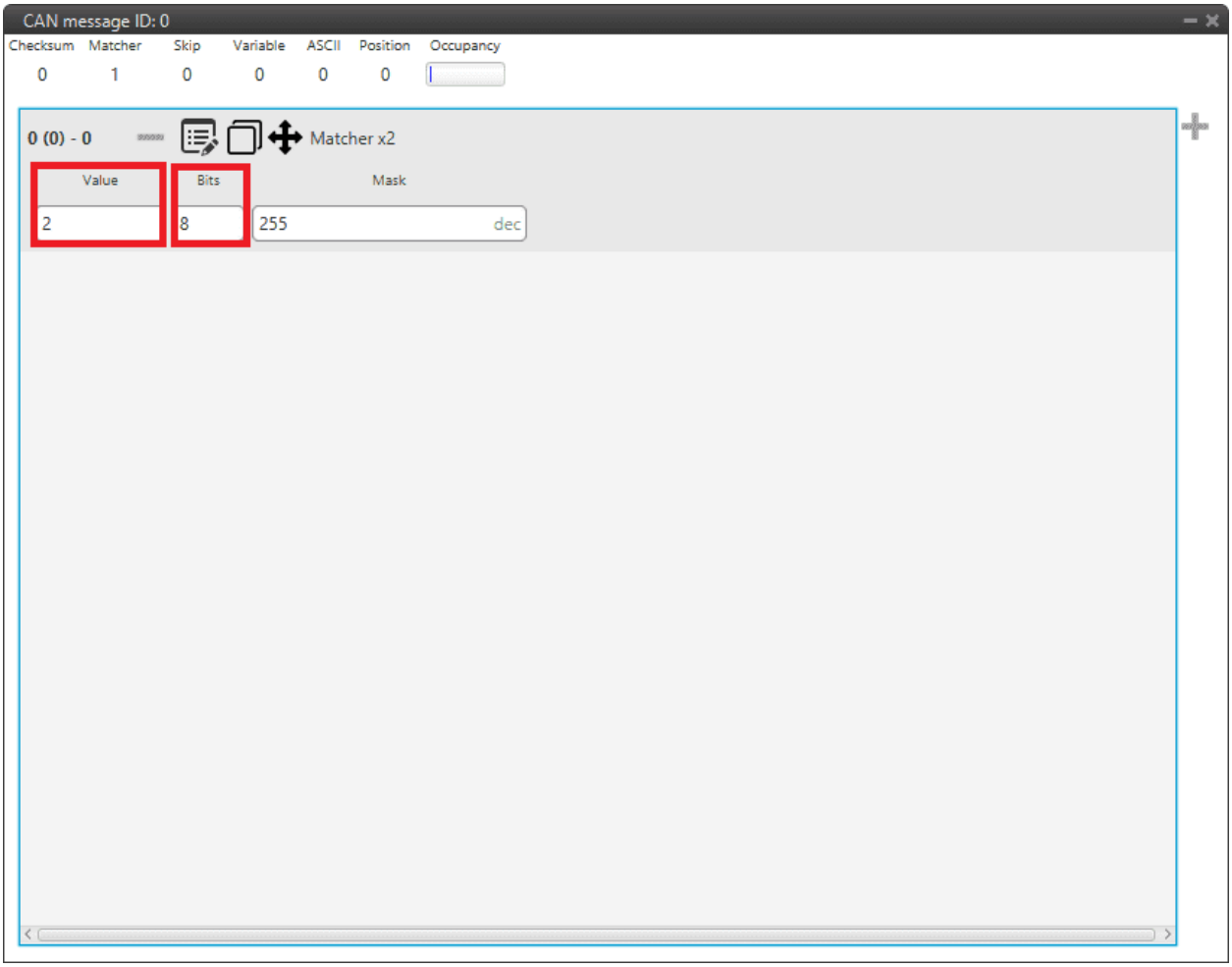


Fig. 1: **Message Type example**

- **Value**: **2**. This is because it is the value for the message for PWMs 0 to 3 (it is **indifferent to the PWM number**).
- **Bits**: **8**. This is because the *Message Type* is an 8-bit message.

2. **data (up to 56 bits - 8 bytes)**: The following bytes refer to the **Message data**.

Next sections decribe each one of the possible messages with an example. The following examples include complete messages, so each beginning corresponds to *Message Type*.

3.1 CEX Status

CEX status message is composed as follows:

Type	Value	Bytes	Description
cmd (t_version)	1	1	Version request / response
data	-	1	Version - Major
data	-	1	Version - Minor
data	-	1	Version - Revision
data (sysaddr)	-	1	Serial number - address 0
data (sysaddr)	-	1	Serial number - address 1
data	-	1 bit	System Error bit (ID 7)
data (CEX status)	-	1 bit	System power up bit error bit (ID 12)
data (CEX status)	-	1 bit	PDI error bit (ID 9)
data (CEX status)	-	1 bit	Memory Allocationbit (ID 8)
data (CEX status)	-	1 bit	File system error bit (ID 6)
data (CEX status)	-	1 bit	CAN A ERROR bit (ID 73)
data (CEX status)	-	1 bit	CAN B ERROR bit (ID 74)
data (CEX status)	-	1 bit	false
data (CEX status)	-	1 bit	Arbiter enabled
data (CEX status)	-	1 bit	Arbiter status

3.2 Arbitration

CEX Arbitration Status message is composed as follows:

- **Message 1**: Sent when "**Send status**" is enabled

Type	Value	Bytes	Description
cmd (t_arbitration)	0	1	Arbitration message
Flag	255 ([0xFF])	1	Status Flag
CAP	-	7 bits	Active Autopilot (Current)
data	-	1 bit	Arbitrating
data	-	1 bit	AP0 Alive
data	-	1 bit	AP1 Alive
data	-	1 bit	AP2 Alive
data	-	1 bit	AP3 Alive (External)
data	-	1 bit	AP0 Ready
data	-	1 bit	AP1 Ready
data	-	1 bit	AP2 Ready
data	-	1 bit	AP3 Ready (External)
data (CEX status)	-	1 bit	System bit error (ID 7)
data (CEX status)	-	1 bit	System power up bit error (ID 12)
data (CEX status)	-	1 bit	PDI bit error (ID 9)
data (CEX status)	-	1 bit	Memory Allocation bit (ID 8)
data (CEX status)	-	1 bit	File system bit error (ID 6)
data (CEX status)	-	1 bit	CAN A bit error (ID 73)
data (CEX status)	-	1 bit	CAN B bit error (ID 74)
data (CEX status)	-	1 bit	false
data (CEX status)	-	1 bit	Arbiter enabled
data (CEX status)	-	1 bit	Arbiter status

- **Message 2** (One for each Veronte Autopilot 1x): Sent when "**Send score**" is enabled

Type	Value	Bytes	Description
cmd (t_arbitration)	0	1	Arbitration message
data	-	1	Autopilot ID [0, 3]
data	-	4 (32 bits)	Autopilot score as Float

3.3 Command PWMs

Each PWM in CEX has to be associated to a Sub Id that indicates which of the CAN Bus message's PWM is listening to.

That allows to control up to four PWMs using the same message if that is desired. Each message is composed by 4 PWMs maximum.

- PWMs from 0 to 3 are sent in a message that includes 4 PWMs coded as 12-bit integers:

Type	Value	Bytes	Description
cmd (t_pwm_0_3_set)	2	1	PWMs 0 to 3
data (pwm0)	-	12 bits	PWM value for sub-id 0
data (pwm1)	-	12 bits	PWM value for sub-id 1
data (pwm2)	-	12 bits	PWM value for sub-id 2
data (pwm3)	-	12 bits	PWM value for sub-id 3

- PWMs from 4 to 7 are sent in a message that includes 4 PWMs coded as 12-bit integers:

Type	Value	Bytes	Description
cmd (t_pwm_4_7_set)	3	1	PWMs 4 to 7
data (pwm0)	-	12 bits	PWM value for sub-id 4
data (pwm1)	-	12 bits	PWM value for sub-id 5
data (pwm2)	-	12 bits	PWM value for sub-id 6
data (pwm3)	-	12 bits	PWM value for sub-id 7

- PWMs from 8 to 11 are sent in a message that includes 4 PWMs coded as 12-bit integers:

Type	Value	Bytes	Description
cmd (t_pwm_8_11_set)	11	1	PWMs 8 to 11
data (pwm0)	-	12 bits	PWM value for sub-id 8
data (pwm1)	-	12 bits	PWM value for sub-id 9
data (pwm2)	-	12 bits	PWM value for sub-id 10
data (pwm3)	-	12 bits	PWM value for sub-id 11

- PWMs from 12 to 15 are sent in a message that includes 4 PWMs coded as 12-bit integers:

Type	Value	Bytes	Description
cmd (t_pwm_12_15_set)	12	1	PWMs 12 to 15
data (pwm0)	-	12 bits	PWM value for sub-id 12
data (pwm1)	-	12 bits	PWM value for sub-id 13
data (pwm2)	-	12 bits	PWM value for sub-id 14
data (pwm3)	-	12 bits	PWM value for sub-id 15

- PWMs from 16 to 19 are sent in a message that includes 4 PWMs coded as 12-bit integers:

Type	Value	Bytes	Description
cmd (t_pwm_16_19_set)	13	1	PWMs 16 to 19
data (pwm0)	-	12 bits	PWM value for sub-id 16
data (pwm1)	-	12 bits	PWM value for sub-id 17
data (pwm2)	-	12 bits	PWM value for sub-id 18
data (pwm3)	-	12 bits	PWM value for sub-id 19

A complete example of how to command PWMs from Veronte Autopilot 1x and read them into CEX can be consulted in the Commanding/Reading PWMs - Integration examples section of the **CEX PDI Builder** user manual.

3.4 Lift MCU telemetry

3.4.1 CEX to 1x

The telemetry sent by CEX via CAN Bus is composed by:

- **Message 1**:

Type	Value	Bytes	Description
cmd (t_mcu_tm1)	18	1	Lift MCU telemetry data 1
data	-	1	Battery Serial Number [0]
data	-	1	Battery Serial Number [1]
data	-	1	Battery Temperature (as received from MCU)
data	-	1	Low Cell Voltage (as received from MCU)
	-	4 bits	Reserved (Zeros)
data (Status Bit)	-	1 bit	PWM receiving Ok
data (Status Bit)	-	1 bit	CAN PWM receiving Ok
data (Status Bit)	-	1 bit	CAN B receiving
data (Status Bit)	-	1 bit	CAN A receiving

- **Message 2**:

Type	Value	Bytes	Description
cmd (t_mcu_tm2)	19	1	Lift MCU telemetry data 2
data	-	1	Battery Serial Number [2]
data	-	1	Battery Serial Number [3]
data	-	1	Battery Serial Number [4]
data	-	1	Battery Serial Number [5]
data	-	1	Battery Serial Number [6]
data	-	1	Battery Serial Number [7]

3.4.2 1x to CEX

The telemetry sent from 1x to CEX must be configured as follows:

Type	Value	Bytes	Description
cmd (t_mcu_cmd)	10	1	Lift MCU battery command
data	-	1	SUB-id A
data	-	1	LED Value A
data	-	1	SUB-id B
data	-	1	LED Value B
data	-	1	SUB-id C
data	-	1	LED Value C

Each CEX will use the SUB-id of the PWM associated to the “Scorpion Tribunus”/PWM ID to identify the value to be used.

3.5 Scorpion Tribunus ESC Telemetry (Lift)

The telemetry read from the Scorpion ESC is sent as:

Type	Value	Bytes	Description
cmd (t_esc_tm)	5	1	Scorpion Tribunus ESC telemetry data
data	-	1	Input voltage in range [0, 85]
data	-	1	Temperature in Celsius
data	-	1	Error Flags from the ESC
data	-	1	Current in Amps [0, 255]
data	-	1	Consumption in mAmps [0, 25500]
data	-	1	RPMs [0, 25500]
data	-	1	Throttle as percentage*2 [0, 200]

3.6 JetiTM ESC Telemetry

The telemetry read from Jeti-TM compatible ESCs is sent as:

Type	Value	Bytes	Description
cmd (t_esc_tm2)	6	1	Jeti ESC telemetry data
data	-	1	Throttle value [0, 200]
data	-	2	Current RPMs
data	-	10 bits	Input voltage in the range [0, 70] Volts
data	-	10 bits	Temperature in the range [0, 575] Kelvin
data	-	12 bits	Current in the range [0, 400.0] Amps

3.7 Jeti BEC Telemetry

The telemetry read from a BEC will be sent in 2 different messages:

- **Message 1**:

Type	Value	Bytes	Description
cmd (t_bec_tm1)	7	1	Jeti BEC telemetry data 1
data	-	2	Device ID
data	-	12 bits	Input voltage in the range [0, 70] Volts
data	-	12 bits	Output voltage in the range [0, 70] Volts
data	-	12 bits	Temperature in the range [0, 575] Kelvin

- **Message 2**:

Type	Value	Bytes	Description
cmd (t_bec_tm2)	8	1	Jeti BEC telemetry data 2
data	-	2	Device ID
data	-	12 bits	Current in the range [0, 100.0] Amps

3.8 Jeti Temperature Sensor Telemetry

The telemetry read from a Temperature sensor will be sent as:

Type	Value	Bytes	Description
cmd (t_temp_tm)	9	1	Jeti Temperature sensor telemetry data
data	-	2	Device ID
data	-	12 bits	Measured temperature 1 in the range [0, 750] Kelvin
data	-	12 bits	Measured temperature 2 in the range [0, 750] Kelvin

3.9 Set Maintenance Mode Command

This command will configure the CEX in maintence mode, setting its configuration in a way that communications can work through SCI-A, SCI-B or Serial-to-CAN configured as:

- **SCI-A** and **SCI-B**: 115200 bauds, 8 data bits, 1 stop, no parity.
- **Serial to CAN**:
  - TX Id: 1301
  - RX Id: 1301

The format of the command is:

Type	Value	Bytes	Description
cmd (t_cmd_maint)	16	1	Command to go to Maintenance Mode

3.10 Stick Selection Command

This command is used to **enable or disable the CEX PPM reader**. If the **address** received matches the CEX's one, CEX PPM reader will be enabled, otherwise it will be disabled.

The format of the command is:

Type	Value	Bytes	Description
cmd (t_stick_sel)	17	1	Jeti Temperature sensor telemetry data
data (sysaddr)	-	1	address 0
data (sysaddr)	-	1	address 1